

PATENT

Atty. Dkt. No. ROC920010221US1
MPS Ref. No.: IBMK10221

IN THE CLAIMS:

The claims remain as follows:

1. (Previously Presented) A method of debugging an application in a debugging environment comprising the application and a debugger program, the method comprising:

automatically counting a number of times each breakpoint in the application is encountered while the application is executing during a test run, wherein the number is increasing and, at any time during the test run, always reflects a current number of times a given breakpoint has been encountered during the test run and wherein counting the number is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

storing the number for each breakpoint in a memory space for use in a subsequent run.

2. (Original) The method of claim 1, further comprising, while the application is stopped, receiving a user-input request to uninterruptedly execute the application until a user-specified breakpoint is encountered some number of times, N-X, where N is a stored number of times the user-specified breakpoint was encountered during the test run and X is a value equal to or greater than zero.

3. (Original) The method of claim 1, further comprising, after the application is stopped at a location in response to a last breakpoint encounter of a particular breakpoint encountered Y number of times at the last breakpoint encounter, receiving a user-input request to uninterruptedly execute the application until the application is again stopped at the location in response to encountering the particular breakpoint Y number of times.

4. (Original) The method of claim 1, wherein automatically counting comprises, for each breakpoint:

Page 2

374368_1

PATENT

Atty. Dkt. No. ROC920010221US1
MPS Ref. No.: IBMK10221

incrementing a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and
resetting the breakpoint-specific counter each time the code segment is entered.

5. (Previously Presented) A computer readable medium containing a debug program which, when executed by a computer configured with an application being debugged during a debug session, performs breakpoint counter operations, the debug program comprising:

counting instructions for automatically counting a number of times each breakpoint in the application is encountered, wherein the number is increasing and, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein counting the number is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and
storing instructions for storing the number for each breakpoint in a memory space for use in a subsequent run.

6. (Original) The computer readable medium of claim 5, wherein the debug program further comprises, execution instructions for uninterruptedly executing the application until a user-specified breakpoint is encountered N-X times, wherein the execution instructions are configured for execution in response to a user request specifying the user-specified breakpoint and a number of encounters N-X, where N is a stored number of times the particular breakpoint was encountered during the test run and X is a value equal to or greater than zero.

7. (Original) The computer readable medium of claim 5, wherein the debug program further comprises, execution instructions for, after the application is stopped at a last stopped location of the test run in response to encountering a last encountered breakpoint Y number of times, processing a user-input request to uninterruptedly execute the application until the application is again stopped at the location in response to again encountering the last encountered breakpoint Y number of times.

PATENT

Atty. Dkt. No. ROC920010221US1

MPS Ref. No.: IBMK10221

8. (Original) The computer readable medium of claim 5, wherein the counting instructions, when executed:

Increment a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and

reset the breakpoint-specific counter each time the particular code segment is entered.

9. (Previously Presented) A method of debugging an application in a debugging environment comprising the application and a debugger program, the method comprising:

associating a counter with a breakpoint and with at least one application code segment in which the breakpoint is located;

incrementing the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

resetting the counter each time the application code segment is entered.

10. (Original) The method of claim 9, further comprising storing a counter value of the counter for use in a subsequent execution of the application.

11. (Original) The method of claim 9, wherein resetting the counter comprises firing an internal breakpoint which does not call a user interface.

12. (Original) The method of claim 9, further comprising:

determining whether a counter value of the counter has reached a user-specified value; and

if so, halting execution of the application and issuing a user notification indicative of the counter value.

PATENT

Atty. Dkt. No. ROC920010221US1
MPS Ref. No.: IBMK10221

13. (Original) The method of claim 9, further comprising:
uninterruptedly executing the application until a counter value of the counter has reached a user-specified value; and then
halting execution of the application.
14. (Previously Presented) A computer readable medium containing a debug program which, when executed by a computer configured with an application being debugged during a debug session, performs breakpoint counter operations, the debug program comprising counter instructions which, when executed:
associate a counter with a breakpoint and with at least one application code segment in which the breakpoint is located;
Increment the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and
reset the counter each time the application code segment is entered.
15. (Original) The computer readable medium of claim 14, wherein the code segment is one of a routine and a loop.
16. (Original) The computer readable medium of claim 14, wherein the debug program further comprises execution instructions for uninterruptedly executing the application until a counter value of the counter is equal to a user-specified value.
17. (Original) The computer readable medium of claim 14, wherein the debug program further comprises, storing instructions for storing a counter value of the counter for use in a subsequent execution of the application.
18. (Original) The computer readable medium of claim 14, wherein the debug program further comprises internal breakpoint setting instructions for setting an internal

PATENT

Atty. Dkt. No. ROC920010221US1
MPS Ref. No.: IBMK10221

breakpoint configured to reset the counter without calling a user interface when the internal breakpoint is fired.

19. (Original) The computer readable medium of claim 14, wherein the debug program further comprises counter value instructions for determining whether a counter value of the counter has reached a user-specified value and, if so, halting execution of the application and issuing a user notification indicative of the counter value.

20. (Original) The computer readable medium of claim 14, wherein the debug program further comprises counter value instructions for allowing uninterrupted execution of the application until a counter value of the counter has reached a user-specified value and then halting execution of the application.

21. (Previously Presented) A computer system, comprising:
a memory containing content comprising at least a debug program to implement a debug session, a breakpoint table configurable with breakpoint-specific counters and an application for debugging; and

a processor which, when executing at least a portion of the content during the debug session, is configured to:

associate a breakpoint-specific counter with a breakpoint and with at least one application code segment in which the breakpoint is located;

increment the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

reset the counter each time the application code segment is entered.

22. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to store a counter value of

PATENT

Atty. Dkt. No. ROC920010221US1

MPS Ref. No.: IBMK10221

the breakpoint-specific counter in the breakpoint table for use in an execution of the application.

23. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to set an internal breakpoint configured to reset the breakpoint-specific counter without calling a user interface when the internal breakpoint is fired.

24. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to uninterruptedly execute the application until a counter value of the breakpoint-specific counter is equal to a user-specified value.

25. (Original) The computer system of claim 22, wherein the debug program, when executed by the processor, configures the processor to issue a user notification indicative of the counter value.